Software Engineering Institute

Results in Relating Quality Attributes to Acquisition Strategies

Lisa Brownsword
Cecilia Albert
David Carney
Patrick Place

February 2014

TECHNICAL NOTE
CMU/SEI-2013-TN-026

Client Technical Solutions Directorate

http://www.sei.cmu.edu

Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This report was prepared for the
SEI Administrative Agent
AFLCMC/PZM
20 Schilling Circle, Bldg 1305, 3rd floor
Hanscom AFB, MA 01731-2125

DM-0000826

Table of Contents

List of Figures

List of Tables

Acknowledgments

Any research endeavor owes much to those who have gone before and to those who have helped make the current work possible. We would like to express our thanks to those who gave us their time and patiently allowed us to interview them regarding their experiences with large acquisition programs: Michael Bandor, Julie Cohen, John Foreman, Harry Levinson, Michael Phillips, John Robert, Sarah Sheard, Jeffrey Thieret, and Mary Catherine Ward.

We would also like to thank Patricia Oberndorf and Bryce Meyer, who provided us the opportunity to prototype a quality attribute workshop adapted for the acquisition domain. In addition, our thanks go to Len Bass, Paul Clements, Rick Kazman, Tom Merendino, and Jeff Thieret who reviewed our work and provided their insights. And finally, our special thanks to John Foreman, Bill Scherlis, and Kevin Fall for continuing to believe in this work.

Abstract

In the acquisition of a software-intensive system, the relationship between the software architecture and the acquisition strategy is typically not specifically examined. The first phase of our research discovered an initial set of failure patterns that result when these two entities become misaligned. Programs with these failure patterns experienced reduced operational capabilities and effectiveness, cost overruns, and significant schedule slips. In other words, these programs resulted in systems failing to satisfy stakeholder needs.

This report describes the conceptual foundations for our project and summarizes the first phase as context for the second phase, which is the major thrust of this report. The current research has centered on demonstrating the existence and utility of acquisition-related quality attributes, embodied in a program's business goals, which then drive the shape of the acquisition strategy. This is comparable to the relationship between mission goals, software-related quality attributes, and the software architecture. This report describes the approach used in phase two to generate 75 acquisition-related quality attribute scenarios based on data derived from more than 23 large government programs spanning business, logistics, command and control, and satellite domains.

1 Overview of the Project

1.1 Introduction

Our project is focused on the relationships between software architecture and acquisition strategy. Although these entities might appear to be unrelated, there is a surprisingly deep connection between them. More specifically, we are concerned with their alignment or misalignment. By identifying and articulating how key entities that are critical to alignment or misalignment interact, we can provide a useful approach for organizations and project managers engaged in acquisition programs.

The key entities of interest are: the architectures themselves, both software and system; the planned acquisition strategy; the quality attributes that drive those architectures and strategies; and the goals (both business and mission) of all of the stakeholders. By examining these entities, we seek to pinpoint major sources that tend either to keep the software architecture and acquisition strategy in harmony or to pull them apart. By so doing, we intend to provide a method for organizations and project managers to avoid patterns of failures that we have discovered, and which are described more fully in Section 2. We expect to validate the utility of this method through pilot applications on projects and programs outside the Software Engineering Institute (SEI).

This project is expected to take place over three phases. Section 2 of this report describes the conceptual foundations for our project and summarizes the first phase, which is documented in "Isolating Patterns of Failure in Department of Defense Acquisition" [Brownsword 2013]. Section 3 discusses our current work for the second phase, which is the major thrust of this report. In Section 4, we describe our plans for the third phase.

1.2 Terminology Used for this Report

Throughout this report, we use the following terms with these definitions.

A *mission goal* is an expression of some operational objective (sometimes referred to a mission driver) and is focused on what the solution should do or how it should behave. These may be described in a number of formal documents in a given acquisition program, or in other, looser ways.

A *business goal* is an expression of some organizational (e.g., Air Force) objective[1] (sometimes referred to as a business driver), not specific to the solution, but focused on what the acquisition (development or maintenance) organization should do or how it should behave. For example, a business goal might refer to budgets, or regulations or policies, or the state of the industrial base. Some of the business goals will be documented in one or more policy documents while others may exist but be unstated. These unstated business goals pose a major difficulty for our research.

Quality attributes are properties of a system. We use the definition from *Software Architecture in Practice*, 3rd Edition [Bass 2012]:

[1] Different organizations (e.g., acquisition program office, contractor, or logistics) involved in an acquisition will likely have different goals and priorities of those goals.

> *A quality attribute is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders. You can think of a quality attribute as measuring the "goodness" of a product along some dimension of interest to a stakeholder.*

Of particular interest for this project are the quality attributes of the system's software, or *software quality attributes*.

Acquisition quality attributes are properties of the program. To paraphrase the definition for quality attributes above,

> *An acquisition quality attribute is a measure or testable property of a program's strategy that is used to indicate how well the program satisfies the needs of its stakeholders. You can think of an acquisition quality attribute as measuring the "goodness" of a program's strategy along some dimension of interest to a stakeholder.*

Acquisition quality attributes are derived from the organization's business goals and relate the business goals to the acquisition strategy in the same way that the software quality attributes relate the mission goals to the software architecture. That is, to formulate acquisition quality attributes for a program implies explicitly developing an acquisition strategy that accommodates them: they must be designed into the strategy to minimize the risk of failure.

A frequently used, and defined, term that is crucial for this paper is *software architecture*; we will adopt another definition from *Software Architecture in Practice* [Bass 2012], where we find:

> *The software architecture of a system is the set of structures of the system needed to reason about the system, which comprise software elements, relationships among them, and properties of both.*

The definition of software architecture given above could be used almost directly for s*ystem architecture*.[2] Indeed, while the reference above does not define system architecture, we can extend the above definition to:

> *The system architecture of a system is the set of structures of the system, which comprise software and hardware elements, relationships among them, and properties of both.*

Both system and software architectures are strongly related to quality attributes. However, it is important to note that the system architect is usually concerned with different quality attributes than the software architect; and sometimes the same qualities will be discussed by system and software architects but with different emphasis (expressed with different scenarios) [Klein 2010].

An *acquisition strategy* is defined by the Defense Acquisition University [DAU 2011] as

> *A business and technical management approach designed to achieve program objectives within the resource constraints imposed. It is the framework for planning, directing, contracting for, and managing a program. It provides a master schedule for research, development, test, production, fielding, modification, postproduction management, and other activities essential for program success. The acquisition strategy is the basis for formulating*

---

[2] Numerous definitions of system architecture exist, with no single definition universally accepted. For our purposes in this report, the version noted above is sufficient. Other definitions can be found in sources such as [IEEE 1998, OPF 2009, Firesmith 2008]

functional plans and strategies (e.g., Test and Evaluation Master Plan (TEMP), Acquisition Plan (AP), competition, systems engineering, etc.)

In examining a given acquisition strategy, we will seek to determine whether it embraces key business goals, as defined above, whether stated or unstated, that can affect the programmatic elements of an acquisition. We refer not only to the requisite program document or documents that bear that title, but also to the wide array of desires, goals, and objectives held by a broad cross-section of stakeholders spanning the hierarchy from very senior to junior.

1.3 Hypotheses of this Work

Our primary hypothesis is that a mismatch between acquisition strategy and software architecture contributes to significant problems in acquisition programs. If a program can avoid the patterns of failure (such as those we identified in phase one of our research), their acquisition strategy and software architecture can be aligned and a program can increase the likelihood of program success.

This hypothesis depends on two key premises. First, a software architecture and an acquisition strategy are necessarily related [Conway 1968, MacCormack 2011, Blanchette 2010, Charette 2003]. These entities form two conceptual structures that are parallel, though in different spheres of the acquisition space (i.e., the software architecture, and the mission users and goals on one hand; and the acquisition strategy, and business stakeholders and goals on the other hand).

Second, the quality of the relationship between software architecture (and related mission goals and quality attributes) and program acquisition strategy (and related business goals and acquisition quality attributes) is of critical importance to the success of the program.[3] This relationship must be one wherein these two entities are both aligned and mutually constraining.[4]

Through our research, we have evolved and refined this initial primary hypothesis and assertions into the following set of derived hypotheses, which together guide our work across the phases of this project:

1. One or more of seven patterns of failing behavior, or "anti-patterns" are major contributors to misalignment between acquisition strategy and software architecture. As summarized in Section 2, our data in phase one made clear that the misalignment was a culprit in several expensive programmatic failures.
2. Business goals that represent the full range of program stakeholders can be explicitly defined and prioritized.[5]
3. Business goals imply acquisition quality attributes that can be defined by program-specific acquisition quality attribute scenarios that can be used to judge the effectiveness of the acquisition strategy—analogous to mission goals expressed in program-specific system and

[3] Within the information systems arena, the relationship of business goals and information technology is termed business-IT alignment, and is considered crucial to the success of an enterprise. For more information, readers can refer to [Strassman 1998, Henderson 1993].

[4] For more information, readers can refer to Figure 1 and [Brownsword 2013].

[5] This is an area of particular emphasis for our project. While business goals can theoretically be defined, in practice a program may choose to leave a goal undocumented, such as for political reasons.

software quality attribute scenarios that can be used to judge the effectiveness of the system and software architecture.

4. Acquisition quality attribute scenarios can be expressed in scenarios parallel to those defined for system/software quality attributes.
5. The Quality Attribute Workshop (QAW), suitability extended, is a reasonable vehicle for collecting a useful set of program-specific acquisition quality attribute scenarios.
6. Using acquisition quality attribute scenarios will yield better acquisition strategies.
7. Reconciling acquisition quality attributes and system/software quality attributes will lead to aligning and mutually constraining acquisition strategy and software architecture.
8. A method that reconciles acquisition quality attributes and system/software quality attributes will be useful to a program.
9. The quality of the relationship between software architecture and acquisition strategy can be objectively measured (in a technique we will develop).
10. Analysis of acquisition quality attribute scenarios can lead to an understanding of a set of significant and pervasive acquisition quality attributes that can lead to a set of acquisition tactics that can be used to successfully address them.

1.4 Demonstration of Success of this Work

This project will have succeeded if the methods for alignment that we are creating can be shown to improve the probability of success in actual acquisition programs. However, this is far from a straightforward task—there are few ways that success can be convincingly attributable to any one acquisition activity (i.e., such as ensuring adequate alignment between architecture and acquisition strategy). In that sense, we face the same problem as do such technologies as risk management.

For example, if some program contains a strong risk management component, and the program then attains a high degree of success, can the risk management element be claimed to be the cause of success? Most observers would disagree; simply having a risk management component of a program cannot be asserted as the root cause of success. Yet given a program with no risk management, and a subsequent disaster, the *lack* of risk management is invariably cited, during the post mortem to discover what went wrong, as a cause of failure.

Another confounding issue with measuring success is that the time lag between when the alignment method would be used (i.e., relatively early in a program) and when program success would be judged (i.e., after IOC, or even later) can be very long. Over such time periods, program personnel change and the users of an alignment method may well have disappeared from the program.

We, however, argue that there are indeed indicators of the value of our work that are sufficiently persuasive. First, the value of an alignment method can be indicated through its users' experience and testimony. A case study approach [Yin 2009, Stake 1995, Baxter 2008] that captures direct experience both with using the alignment method and observing how it impacts the ongoing acquisition process is a very strong indicator of the method's efficacy, particularly if the users have had experience with previous acquisitions. Being able to compare several experiences can lead, if not to specific metrics, at least to assurance that the method has positive value.

Second, our approach is an analog to the software-based QAW. At a software-based QAW, diverse stakeholders are brought together to jointly define the desired software quality attributes of the software architecture. There is considerable experience [Barbacci 2001] that such encounters have value: they invariably produce awareness among the participating stakeholders that there are competing, and often conflicting priorities for the architecture. As with other aspects of software, unearthing potential conflicts early, e.g., before the architecture has fully been defined, produces a significant saving over changing it once it has been implemented. (The savings, though, are extremely difficult to quantify.) Therefore performing the same service with the various business-focused stakeholders of an acquisition strategy should have comparable (and equally difficult to quantify) value.

Notwithstanding the difficulties, in discussions with acquisition knowledgeable people about our research, they have uniformly expressed their belief that finding ways to improve the alignment between an acquisition strategy and a software architecture will prove a significant and valuable technology.

2 Background

2.1 Foundations of this Work

This research builds on significant previous work. There are three foundations that frame our research. The first is our recognition and appreciation of the considerable body of knowledge that has emerged from the SEI's ongoing work in software architecture.[6] In that work, the critical relationships between a system's software architecture and the aggregate collection of its users' goals were studied. Several methods were developed focused on achieving greater consistency between the architecture and the users' goals. Three SEI methods are of particular relevance to our work:

- generating, documenting, and prioritizing a system's quality properties (e.g., performance, availability, interoperability): the Quality Attribute Workshop (QAW) [Barbacci 2003].
- eliciting and documenting high-priority business and mission goals, and capturing the architectural implications of those goals: the Pedigreed Attribute eLicitation Method (PALM) [Clements 2010].
- evaluating architectures and the engineering tradeoffs among possibly conflicting quality goals: the Architecture Tradeoff Analysis Method (ATAM) [Clements 2002].

The second foundation of our research lies in the ongoing efforts by the DoD to improve the acquisition process. These efforts have had two positive effects on our research. First, the business goals of the department have been clearly stated; second, they make the relationship between these business goals and the program's acquisition strategy more explicit. With efforts such as Better Buying Power 2.0, improvements are being sought in delivering better value to both the taxpayer and the warfighter [USD 2012]. The department is looking for changes to how the acquisition programs are structured to better meet the major business goals of the organization:

- improve the affordability of programs
- incentivize greater productivity and innovation in both industry and government
- eliminate unproductive processes and bureaucracy
- promote effective competition
- improve tradecraft in acquisition of services
- improve the professionalism of the total acquisition workforce

A third foundation of our work is the SEI's experience with more than 100 Independent Technical Assessments (or ITAs, and often informally called "red teams"). Such assessments are commissioned by the government to provide third-party analyses of a program's health, quality of progress, and similar conditions. Our team's ITA experiences and those of our colleagues strongly corroborate the observations noted in the paragraph above. More specifically, we have also observed that among the factors leading toward failure, the *acquisition strategy* featured prominent-

[6] Although we specifically call out SEI research in software architecture, we are not limited to this source in our research. We are leveraging other work in the broader architecture and requirements community, particularly as we move into phase three of this project.

ly. We have often found that failing to state the large collection of desires and goals leads to strategies that frequently contain significant internal contradictions that adversely affect programs.

2.2 Phase One: Characterizing Failure Patterns

As previously noted, developing a validated method that facilitates the alignment of a software architecture and the acquisition strategy within a program is a multi-phase project. Our objective for phase one was to discover of the potential causes of mismatch between the acquisition strategy and the software architecture that contribute to acquisition program problems. In this section, we summarize the activities and outcomes of the first phase of our project, covering the following topics:

- patterns of failure, or anti-patterns
- entities and relations that pertain to the anti-patterns
- conclusions we drew from this phase of our research

2.2.1 Anti-Patterns

Our initial focus was on gathering data about the mission and business goals for a program, elements of its acquisition strategy, software and system architectures, quality attributes, key external and internal program events that occurred over the life of program, and program outcomes. To that end, we conducted interviews with SEI personnel who had participated in ITAs of large DoD and other government acquisition programs, all of which had encountered difficulties during their lifecycles.

In analyzing this data, we discovered several recurring patterns that appeared to be connected with mismatches between the acquisition strategy and the software architecture leading to programmatic failures. We based some of this analysis on an existing body of research on design patterns:

> *[A pattern] describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice; in other words, a pattern is a template that can be used in a specific situations [Alexander 1977].*

We transposed this description somewhat, since Alexander's description of a pattern includes the presence of a solution to a problem, while we were describing only the problem element. This transposition is commonly called an *"anti-pattern"* within the software community [Brown 1998].

While there are many patterns of failure in acquisitions, the analysis of our data identified a number of anti-patterns that were evident in the programs we studied. These were

1. Undocumented Business Goals: the lack of well-documented business goals expressed as they apply to an acquisition program
2. Unresolved Conflicting Goals: the lack of analysis and reconciliation of known goals (mission or business)
3. Failure to Adapt: failure of an acquisition program to modify the architecture and the acquisition strategy in response to changing goals, priorities, or technology

4. Turbulent Acquisition Environment: requested changes are so frequent and contradictory that an acquisition program cannot realistically accommodate them
5. Poor Consideration of Software: critical decisions made early in an acquisition program's lifecycle have strong negative implications on the system's software
6. Inappropriate Acquisition Strategies: the acquisition strategy fails to consider important software attributes
7. Overlooking Quality Attributes: a failure to define and use quality attributes in the definition of the software architecture or acquisition strategy

For each of these anti-patterns, we described the context in which the problem usually emerged, the specific nature of the problem, the observed response to the problem (NB: not a solution, but rather the observed response that failed to solve the problem), and examples of the consequences, both immediate and long-range. Brownsword provides further descriptions of each of the anti-patterns [Brownsword 2013]. It is important to note that we are not asserting that observed anti-patterns are the only negative influence on an acquisition. We recognize that other forces can also affect a program and its potential success.

2.2.2 Entities and Relations that Pertain to Anti-Patterns

Based on our analysis, we conjectured that there were a small number of critical entities involved in these anti-patterns, and that they were related in significant ways. The entities are:

- mission goals, and the (system and software) quality attributes implicit in those goals
- business goals, and the (acquisition) quality attributes implicit in those goals
- the acquisition strategy
- the software and system architectures, which are closely related, but separate
- the different sets of stakeholders who have expressed needs that are captured by the mission and business goals

The set of entities and relations is shown in Figure 1 and is at the heart of our primary hypotheses: if these relationships between the main entities of an acquisition are strong, then there is a higher chance that the acquisition strategy and the software architecture are mutually constraining, and at least this cause of acquisition failure can be avoided. For example, by strengthening the relationship "stakeholders have business goals," such that these goals from the salient stakeholders are collected and exist in a coherent artifact, then the anti-pattern #1 (Undocumented Business Goals) would not occur, or be substantially reduced. Brownsword discusses further how the anti-patterns noted above are affected by these relationships [Brownsword 2013].

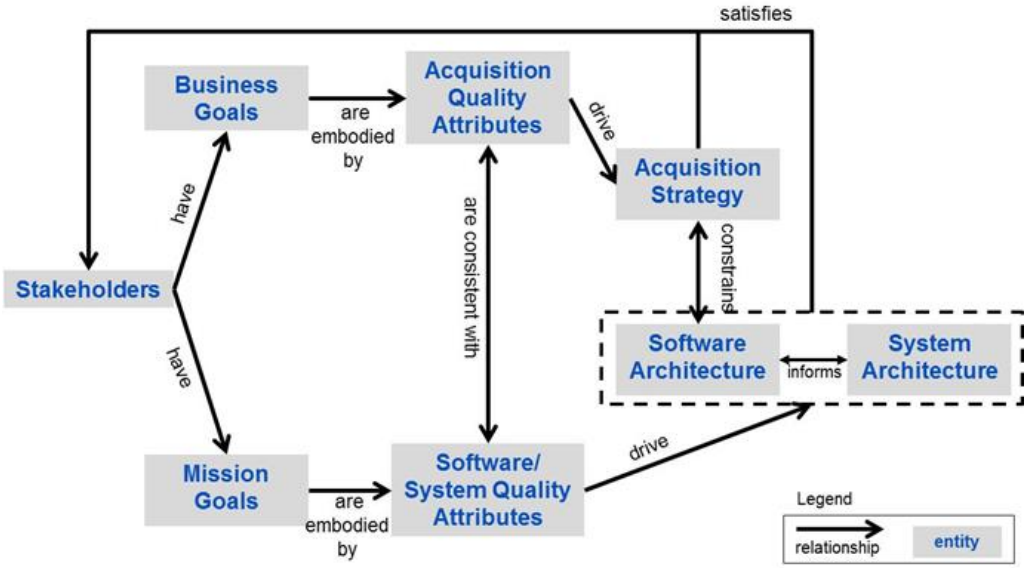

Figure 1: Desired Relationships Among the Principal Entities

2.2.3 Conclusions from our Phase One Research

As shown in Figure 1, the business goals for a program are a key entity. Although our data showed that a number of important stakeholders have business goals, these goals are often not expressed or captured. Further, we observed that there was no process for doing so. Without such a process, it is difficult to analyze these goals for conflicts with other mission or business goals, let alone to analyze for the sufficiency of the acquisition strategy to accommodate the desired business goals.

Through our phase one research and analysis, we concluded that the business goals, similar to mission goals, will have quality attributes that should be the main drivers for the acquisition strategy. We assert that these acquisition strategy-related quality attributes are as important as those derived from the mission goals and refer to them as *acquisition quality attributes*. We posit that these acquisition quality attributes are a critical means for forming and analyzing the acquisition strategy for a particular program.

How are these acquisition quality attributes best elicited and captured? Can they be used to surface potential conflicts among other business goals? Can they show possible impacts to an acquisition strategy? Exploring these questions became the basis for phase two of our project.

3 Phase Two: Exploring Acquisition Quality Attributes

The focus for phase two of our project was to demonstrate the applicability of acquisition quality attributes. Our premise (from the hypotheses in section 1.3) is twofold: (1) there is a set of program specific acquisition quality attributes that can be derived from a program's business goals that drive its acquisition strategy, and (2) acquisition quality attributes can be expressed in a way that allows them to be analyzed and evaluated. Our research methodology for this phase consisted of the following activities:

- form a list of potential acquisition quality attributes
- define an approach for expressing program-specific acquisition quality attributes that allows them to be effectively reasoned about
- elicit and capture acquisition quality attribute scenarios
- build and validate a prototype workshop to elicit acquisition quality attribute scenarios
- analyze the acquisition quality attribute scenarios

Much of this work followed a similar path as that used with the original research on software quality attributes. In particular, as did the developers of QAW and PALM, we adopted the principle of using scenarios to give precise meaning to acquisition quality attributes.

3.1 Potential Acquisition Quality Attributes

There are many different ways that attributes, whether the software quality attributes of software architecture, or the acquisition quality attributes we are presently focusing on, can be aggregated. For instance in the initial research on software architecture and software quality attributes, we see that the quality attributes of interest tended to group into a fairly small number of very general attributes (e.g., performance, dependability, security), which then broke down into several levels of greater specificity (e.g., performance first can be broken down into latency, throughput, and capacity; latency in turn can be broken down into precedence, criticality).

While it was apparent that acquisition quality attributes will vary in their breadth and generality, we first speculated on what the raw vocabulary of acquisition quality attributes might consist of. To that end, we considered a number of possible areas in which acquisition-related attributes would be found. For example

- contract issues (e.g., legality, contract manageability, comprehensiveness, appropriateness)
- program management issues (e.g., accountability, management visibility)
- program execution issues (e.g., promptness in reporting problems, responsiveness to government requests)

We decided, however, to simply create a list, unordered and without concern to generality or specificity, and use the scenarios to give us insight as to what a reasonable taxonomy might be. The list of roughly 30 acquisition quality attributes is shown in Appendix A. These are largely drawn from DoD acquisition strategy guidance and instructions and then augmented by our research.

As we reflected on the collection of acquisition quality attribute scenarios generated through our research, we saw emerging themes that may provide the basis for a possible taxonomy of acquisition quality attributes in the future. We observed the following:

- The acquisition quality attribute *executability* tended to occur when program cost, schedule, and performance were in balance and could, therefore, be further decomposed into three other acquisition quality attributes:
  - *affordability*, where costs are appropriate for the performance and schedule planned
  - *schedulability*, where schedule is appropriate for the performance and cost planned
  - *performability*, where performance is appropriate for the cost and schedule planned
- The acquisition quality attribute *flexibility* tends to occur when a program can respond appropriately to changes in cost, schedule, or performance.
- The acquisition quality attribute *program survivability* tends to occur where the extent to which a program can defend against external pressures or threats.
- The acquisition quality attribute *realism* tends to occur when stakeholder expectations are compatible with the executing program.
- The acquisition quality attribute *programmatic transparency* tends to occur when a program knows the current balance of cost, schedule, and performance.
- The acquisition quality attribute *innovativeness* may be a form of flexibility.
- The acquisition quality attribute *staffability* may be a form of survivability.

These initial observations are oriented exclusively to acquisition and programmatic factors and do not attempt to account for software architecture decisions. Further work during phase three of this project may refine and extend these early observations.

3.2 Expressing Program-Specific Acquisition Quality Attribute Scenarios

We recall one of our central hypotheses:

> *Business goals imply acquisition quality attributes that can be defined by program-specific acquisition quality attribute scenarios that can be used to judge the effectiveness of the acquisition strategy—analogous to mission goals expressed in program specific system and software quality attribute scenarios that can be used to judge the effectiveness of the system and software architecture (from Section 1.3).*

We next considered how such program-specific scenarios might be constructed. Once again, the example from the work in software architecture-based scenarios was invaluable. In a software architecture QAW, end users are encouraged to create small "stories" that specify some event (the "stimulus") that occurs under particular conditions (the "environment") and then the desired behavior (the "response") of the system.

An example of such a scenario from the domain of software architecture[7] might be the following:

Stimulus:	An internal component fails
Environment:	During normal operation

[7] From Reasoning About Software Quality Attributes, http://www.sei.cmu.edu/architecture/start/reasoning.cfm

Response:	The system is able to recognize a failure of an internal component and has strategies to compensate for the fault

A parallel example from the domain of acquisition might be the following:

Stimulus:	An unexpected budget cut
Environment:	For a multi-segment system
Response:	The program is able to move work between major segments to speed up or slow down separate segments within the available funding

Subsequently, a program would expand these three-part scenarios to six parts: the original three parts; who generates the stimulus (the "source"); the artifact that the stimulus most strongly affects, and the measure(s) by which the success of the response will be evaluated. In practice, this expansion and refinement takes considerable effort. We investigated this refinement and expansion for many of the acquisition quality attribute scenarios created in this phase and we expect to continue our investigations in the following phase of our work. For simplicity of presentation in this report, we use the three-part scenario form.

In developing these scenarios, we discovered the following parallels with the software quality attribute scenarios:

Software Quality Attribute	Acquisition Quality Attribute
Software architecture	Acquisition strategy
System	Program
Architect	Program manager

Otherwise, we preserved the approach of gathering the three-part form of the scenario (stimulus, environment, and response), which could later be refined and expanded into six-part scenarios.

3.3 Elicit and Capture Acquisition Quality Attribute Scenarios

A major component of phase two was the task of collecting and describing a large number of scenarios that would provide us with the necessary basis for analyzing alignment and incompatibilities. We sought in this phase to develop as many scenarios as possible, since we expected to need sufficient raw data for the analysis that would follow.

To accomplish the collection of scenarios, we needed additional data in the form of actual acquisition situations, events that occurred (whether beneficial or otherwise), the types of conditions in which the programs unfolded, the kinds of authority structures and strictures that were present and related kinds of information. To this end, we gathered a large body of actual acquisition experiences from a variety of acquisition professionals, and then refined that experience into a collection of acquisition quality attribute scenarios.

3.3.1 Conduct Investigations in Scenario Collection

The SEI provides a deep pool of acquisition experience, and we made extensive use of that expertise in developing scenarios. The aggregate data covered 23 government programs. We used two approaches:

- interview program management office personnel
- interview Independent Technical Assessment (ITA) members

Interview Program Management Office Personnel

We interviewed senior SEI staff who had been either program managers, had positions of authority, or provided close support to the program management offices. This was an extremely valuable body of information, since the program management office is at the heart of any acquisition. We sought information from two different temporal perspectives. First, looking forward in time, what kinds of issues were of concern at the very start of a program, before the acquisition strategy was finalized? And second, during program execution, did unexpected events occur, and how were they accommodated?

Interview ITA Members

We built on and expanded the activity from phase one where we interviewed participants on ITAs that had been performed by the SEI. Such assessments are commissioned by the DoD and other government agencies to provide third-party analyses of a program's health, quality of progress, and similar conditions. Common to all of these ITAs was that, at some point in the program history, there were sufficient problems noted that one or more persons in authority requested an assessment from the SEI to provide an independent review of the program's execution. As with phase one, we found that the findings of the ITAs provided useful material for our research. Specifically, we sought relevant information about each program's acquisition strategy, background of the program, including its scope and motivation, details of the program, e.g., size, timeline, funding, and most critically, the mission and business goals of the program.

3.3.2 Construct Acquisition Quality Attribute Scenarios

The data we collected from the interviews described actual acquisition experiences, each one concerning events with significant effect on the success of a given program. For each of the descriptions, we isolated the event that occurred: this formed the *stimulus* of the scenario. The following are examples of the kinds of stimuli that we noted:

- discovery that a contractor is non-performing due to lack of capability on staff
- schedule is not being met because of poor planning by a subcontractor
- need to react quickly and there are only a limited number of contractors able do the work

We then noted the conditions that were present when that stimulus occurred. By "conditions that were present," we refer to a variety of things that might provide the *environment* for the scenario. Examples included:

- a program where the work is classified and it takes a long time to get people cleared
- shortly after award of a new contact to update the airframe of a legacy program
- warfighters have urgent operational needs for program changes and there is a limited number of contractors able do the work

And finally, we considered the behavior, i.e., the *response* to the event. At this point, our focus became divided. On one hand, some of our data (generally drawn from ITA experiences) indicat-

ed what a program had actually done, which was, in retrospect, failing behavior. By contrast, we also examined data where a program had planned well in its earliest days, and when some unforeseen event occurred, the program responded in a beneficial manner.

Comparing these different programs was at the heart of our work during this phase. Just as in a software-related QAW, where end users try to forecast some events, and the architect realizes that the architecture must be designed to accommodate that event, developing acquisition quality attribute scenarios for a real program would also have acquisition-focused program participants try to forecast comparable events, and the persons defining the acquisition strategy must realize that the strategy must be designed to permit an appropriate and beneficial response.

We therefore cast each scenario in "beneficial" terms. For example, if our data described a program that had not responded well to the stimulus, we anticipated the behavior that would have been more appropriate. Had that program undergone an acquisition quality attribute analysis similar to a software-related QAW, we would presume that the unexpected stimulus was foreseen, and an appropriate response built into the acquisition strategy.

The following are examples of the scenarios we constructed. We added an element to show possible acquisition strategy tactics, that is, how the scenario response could be incorporated into an acquisition strategy. We created these examples based on our research team's acquisition experience. They are provided solely as examples to illustrate the relationship between the scenarios and an acquisition strategy and should not be construed as the only, or a preferred, tactic. Other tactics are certainly possible. Acquisition personnel would need to create actual tactics suitable for their particular program's needs and constraints. In effect, this is parallel to how a software architect would refine the planned architecture based on the results of software-related quality attribute scenarios.

Acquisition Quality Attribute Scenario A:

Stimulus	One associate contractor refuses to share information with other contractors
Environment	Associate contractors are competing on other customer work
Response	Use management structures and incentives to force collaboration
Potential Acquisition Tactic	Create contract requirements so government can monitor collaboration

Acquisition Quality Attribute Scenario B:

Stimulus	A new need arises when we want to react quickly
Environment	There are only a limited number of contractors able to do the work
Response	Work to satisfy the need is added to an existing contract
Potential Acquisition Tactic	Award IDIQ contracts to multiple (perhaps eight or so) vendors and issue task orders in a round-robin fashion

Acquisition Quality Attribute Scenario C:

Stimulus	There is pressure to use non-developmental item (NDI) products
Environment	That are incompatible with the desired architectural strategy of creating an "Apple-like" system
Response	The pressure is ignored and the program continues to follow and enforce its architectural strategy
Potential Acquisition Tactic	Do more upfront work in terms of "market research" so that the approach can be explained and defended against alternatives through the life of the system; this research needs to be done in the context of the enterprise strategy

3.4 Build Prototype Workshop to Elicit Acquisition Quality Attributes

The investigations in scenario elicitation through interviews were focused largely on how to form viable acquisition quality attribute scenarios from the interview data. This was a necessary step toward developing a technique that we could use with a program that was in the process of forming its acquisition strategy. We again leveraged the work in the software architecture community in eliciting software quality attributes, namely the QAW. Where necessary, we made some modifications, but in essence, the prototype *acquisition* QAW, an AQAW, paralleled the QAW closely. The shape of a QAW is as follows:

- Opening presentations define the QAW process, describe the program's business and mission drivers, and outline the plan for the system architecture. The latter two presentations are then used to determine the key architectural drivers which are used as the focus of scenario brainstorming.
- Scenario brainstorming takes places in a round-robin fashion where each workshop participant is, in turn, asked to provide a scenario or pass for the round. Scenarios are provided in a three-part format of stimulus, environment, and response. The goal is to generate as many scenarios reflecting the architectural drivers as quickly as possible.
- The last steps of a QAW relate to analysis of the generated scenarios. The scenarios are consolidated so that duplicates are removed. The remaining scenarios are prioritized, and then refined into six-part scenarios by adding the source of the stimulus, the artifact stimulated, and the response measure.

We adapted the QAW to form an AQAW primarily by placing more emphasis on the business presentation and replacing the architecture presentation with one on the program's acquisition strategy plans. Our approach was to follow essentially the same process though the initial analysis, with one further change. Prior to scenario brainstorming, we modified the identification of architectural drivers step to focus on acquisition strategy drivers instead of the architecture.

We conducted a prototype of the AQAW as a test to determine whether our QAW variant could, indeed, elicit acquisition quality attribute scenarios. The prototype was conducted on a real program using SEI staff that supported the program in place of members of the program office. The program of interest wanted to upgrade a current software release fielded on a complex array of servers that was used by distributed, network-connected users. The program sought to upgrade the system, possibly using some type of cloud-based structure, to meet a larger set of capabilities.

Thus, in the same way that a QAW must recognize business drivers and programmatic constraints, the AQAW had to recognize the impact of the software and system architecture.

In a QAW, stakeholders representative of the user community and the architects participate in the workshop. For an AQAW we would envision stakeholders from the acquisition, user, and engineering communities. To compensate for this breadth of stakeholders, we asked the SEI team members to role play the actual stakeholders associated with the program, identifying which role they were playing.

We did several additional modifications in this prototype to accommodate time constraints of the SEI staff that we would not foresee including in a production setting. These modifications included:

- foregoing the formal business and mission drivers and architecture presentations in favor of a verbal descriptions and sketches on the whiteboard
- focusing on scenario capture with more emphasis on refinement during the capture step than would be desirable in an actual QAW
- performing little analysis during the workshop with subsequent analysis and refinement performed by the AQAW team after the workshop

This prototype provided us with an opportunity to work with SEI staff with intimate knowledge of a real program that had developed its acquisition strategy but use as little of their time as possible while attaining a satisfactory result. The prototype AQAW generated twenty acquisition quality attribute scenarios, which are listed in Appendix C. While only a single case, the prototype has successfully demonstrated that an AQAW is a plausible approach for capturing acquisition quality attribute scenarios. This will be an area of further investigation in phase three of our project. In the prototype AQAW, similar to the architecture-focused QAW, the scenarios were identified through brainstorming that is dependent on the specific participants in the workshop. In particular, we think it is important to explore a more deterministic approach for eliciting acquisition quality attribute scenarios.

3.5 Analyze Acquisition Quality Attribute Scenarios

Our interviews generated 55 acquisition quality attribute scenarios (a complete list is in Appendix B) in addition to 20 acquisition quality attribute scenarios (listed in Appendix C) captured in the prototype AQAW. We now had sufficient data to begin the task of analysis. There were several steps involved:

- identify possible groups of related scenarios
- perform descriptive analysis
- perform content analysis
- demonstrate how different scenarios result in different acquisition strategies
- find incompatibilities between scenarios (that could imply misalignment)

3.5.1 Identify Possible Groups of Scenarios

The scenarios generated from interviews were developed by asking our interviewees to identify memorable negative and positive events that occurred in the programs they were associated with, i.e., we were identifying possible scenarios *after* the fact. Thus, they gave us scenarios that largely represented dominant problems encountered in their programs. Given the large number of programs represented, there were repeating or at least similar, program events. Similar scenarios were grouped together, forming the following categories:

- Contractor Capability Scenarios: Each highlighted an example where either a specific contractor or the industrial base as a whole did not have the available skills necessary to execute the contract as planned.
- Program Office Capability Scenarios: Each highlighted a stimulus where the program office did or did not have the available skills necessary to execute the selected strategy.
- Sharing Across Programs Scenarios: Each highlighted a circumstance where shared use of components across what had been independent organizations was a significant component of the program strategy.
- Innovative Solution/New Technology Scenarios: Each highlighted a program office seeking innovative/new solutions to their needs.
- Other Software Lifecycle Scenarios: These highlighted other circumstances encountered by a program that did not fall into any of the other four categories. As we continue our research in phase three, this category is likely to be refined and new categories identified.

The distribution of acquisition quality attribute scenarios elicited through the interviews into the scenario categories is shown in Table 1.

Table 1: Distribution of Scenarios Derived from Interviews

Classification	Number
Industrial Base Capability	10
Program Office Capability	16
Sharing Across Programs	12
Innovative Solution/Technology	8
Other Software Lifecycle	9

It is important to note that the scenarios captured in the prototype AQAW differ from those developed from the interviews. In the workshop we worked with a single program and we explicitly elicited and captured scenarios that were driven from their specific business goals. Thus, there is at most two scenarios representative of each of the scenario categories listed above.

3.5.2 Perform Descriptive Analysis

The interviews of former program management office personnel and ITA team member covered more than 23 large government programs spanning business, logistics, command and control, and satellite domains. Through the interviews and the AQAW we collected more than 75 scenarios. We would not expect these scenarios to all be present in a single program. Likewise, given the number of programs represented, we found several scenarios that were similar. There were a number of observable trends that are noteworthy:

- frequency of the acquisition quality attributes that the scenarios defined
- commonality of themes
- commonality of acquisition solutions

Frequency of Acquisition Quality Attributes Defined by the Scenarios

As part of capturing each scenario, we associated it with the acquisition quality attribute it defines. A few scenarios were ambiguous and could support two different acquisition quality attributes. These were counted twice. Table 2 shows the frequency of the acquisition quality attributes.

Table 2: Frequency Count of Acquisition Quality Attribute Scenarios

Acquisition Quality Attribute	Frequency
Flexibility	23
Performability	15
Realism	14
Affordability	10
Survivability	6
Executability	5
Responsiveness	4
Programmatic Transparency	2
Innovativeness	1
Schedulability	1

As Table 2 indicates, flexibility, performability, realism, and affordability are the four most prevalent acquisition quality attributes from the scenarios collected so far. In some ways, this is expected since an examination of these scenarios shows that each of them is focused on ensuring that the acquisition completes successfully. This means that the acquisition strategy has to be flexible, realistic, and affordable in order to survive the random events that occur in any long-lived acquisition. In addition, all players must be able to perform their tasks successfully, for the same reason.

Commonality of Themes

The two most common themes occurring in the scenarios relate to personnel and requirements as part of either the stimulus or the environment: eleven (20%) of the scenarios reference lack of skilled personnel in either the program office or the contractor and seven (13%) of the scenarios reference the reality of changing or urgent requirements.

Commonality of Acquisition Solutions

While the broad number of solutions tended to be specific to the program and issue, there were two solutions that frequently emerged:

- use pre-awarded contracts in one form or another
- run things in parallel and do a down select

It is possible that, with further analysis and data, the above two solutions will form the basis for clusters of acquisition strategy tactics, much as is the case in the domain of software architecture.

3.5.3 Perform Analysis of Content

Just as for software quality attributes, the general form of an acquisition quality attribute scenario can be expressed as "if this event occurs (stimulus) when we are in this state (environment) then we want to be able to do this (response)." However, if we examine the acquisition quality attribute scenarios we have collected and focus on the stimulus, we see that the majority of these stimuli follow a slightly different form. Specifically, the stimulus itself is in two parts, where the first part reveals an issue with one of the three major programmatic controls (cost, schedule, and performance) and the second part defines the reason for that issue.

For example, scenario 4 (as listed in Appendix B) has the stimulus "*The schedule is not being met because of poor planning by a subcontractor*." We can see that the former part is that the schedule is not being met, and then a reason is given as the latter part. It is logical that most scenarios will have a stimulus of this form, since cost, schedule, and performance are key indicators of the acquisition's progress and are the areas on which a program reports. However, if cost, schedule, and performance were the only pieces of the stimulus, it would be impossible to fashion a detailed response. Thus, a reason for the perturbation is also a necessary part of the stimulus. The response can then be crafted to mitigate the reason; in the case of this scenario, one of the responses was "*the prime contractor trains the subcontractor in project management.*"

Just as occurs in a QAW, different acquisition quality attribute scenarios from a single program can be variants of each other. In the case of software quality attributes, these variants are frequently based on the same stimulus in different environments that could lead to a different response. We have found in the case of acquisition quality attributes, these variants are more likely to be based on different responses to the same stimulus and the same environment.

In hindsight, the different responses in acquisition quality attribute scenarios are a reflection of the nature of acquisition. Acquisition is about people, not software, making decisions; and, frequently, these decisions are strongly influenced by factors outside the control of the program. System and software responses are more deterministic.

The external environmental influences on acquisition decision are very difficult—maybe even impossible—to explicitly delineate. The number of factors that could influence the decisions a program manager makes are numerous, and can range from obvious factors, such as the effect of a unforeseen budget cut or new direction on the schedule driven by operational crisis, to hidden or subtle factors such as the relationship between the program manager and the customer organization.

3.5.4 Different Scenarios Result in Different Acquisition Strategies

If acquisition quality attribute scenarios are truly analogous to software quality attribute scenarios, then we should be able to anticipate the influence from the acquisition quality attribute scenarios on the "goodness" of the acquisition strategy analogous to the way software quality attribute scenarios influence the "goodness" of the software architecture. Even if applied after the acquisition strategy has been developed, we should be able to use the acquisition quality attribute scenarios to distinguish between acquisition strategies or to determine the appropriateness[8] of the acquisition

[8] We avoid the judgmental terms "good" and "bad" since most strategies will be "good" with respect to some scenarios and "bad" for others. A "good" strategy is one that is appropriate for the crucial scenarios.

strategy with respect to any given scenario. Since these two ways to use an acquisition quality attribute scenario are simply a matter of timing, we will focus on the first use with the knowledge that if we can demonstrate a scenario might influence the acquisition strategy, then we can also use a scenario to test a strategy.

For an acquisition quality attribute scenario to have an influence on the acquisition there must be some element of the scenario that leads the program office to make some kind of choice between one strategy and another. Examining the relationship, we see that the acquisition strategy should be such that the program office can make the response specified in the acquisition quality attribute scenario. Thus, if there are to be different scenarios, it is reasonable to see that either we have two different scenarios with different responses or a single scenario that leads to two different responses. In either case, we show that different scenarios lead to different acquisition strategies.

Examining the scenarios we collected (listed in Appendix B), we see a number relating to new technology and the issues that arise if the chosen innovative technology fails to deliver on its promises. From the collected scenarios we may posit a single scenario with two variant responses, where the variation depends on the environment component (indicated by italics) of the scenario:

1. A new technology the program office expects to use is found to be unsuitable *where schedule is of prime importance*; the program office switches to an alternative that is also currently under development and is evaluated to be suitable.
2. A new technology the program office expects to use is found to be unsuitable *where costs must be kept as low as possible*; the program office instructs the contractor to restart but using an alternative technology.

We can see from these two scenarios that the stimulus is the same but the environment changes; in the first case, schedule is more important than cost and the second case reverses their relative importance. In the first case, an acquisition strategy starting multiple developments simultaneously with a requirement for some kind of decision between the alternatives would be appropriate. In the second case, a strategy starting a single development and continuing with that until such time as it was found to be infeasible and then switching to an alternative would be appropriate.

As simple as this example is, it demonstrates that different acquisition quality attribute scenarios can lead to different acquisition strategies. This strengthens our contention that our use of acquisition quality attributes and acquisition quality attribute scenarios is, indeed, analogous to the use of software quality attribute and software quality attribute scenarios and that we may continue to rely on methods and mechanisms developed for that purpose to assist with the creation of sound acquisition strategies.

3.5.5 Find Incompatibilities Between Scenarios

In software, we frequently find that two or more software quality attributes are incompatible with each other (e.g., performance attributes are often in conflict with security attributes) and, thus become the subject of architectural tradeoffs. We, therefore, examined the acquisition quality attribute scenarios to determine the possible kinds of incompatibilities between different scenarios. We first considered incompatibilities that could occur between different acquisition-related scenarios, and then considered incompatibilities that could occur between an acquisition-related scenario and an architecture-related scenario.

Incompatible Acquisition Scenarios

Conflicts between scenarios are not always obvious, and may not become apparent immediately. In the following example, for instance, the conflict is quite subtle without some analysis. Organization ABC has deployed a large, complex legacy system in multiple operational locations, where each location installed its own local variant of the system. Over time, these variants diverged in response to differing requirements of the local users. The various operational locations identified a need to share data in a more integrated way. A new program was initiated to acquire one replacement capability that would support all of the differing needs across the multiple fielded locations. The program decided to implement an incremental approach to replacing the legacy system so they could respond to budgetary constraints and uncertainties.

The operational processes and need vary between the current fielded locations. Understandably the user requirements for the new capability also differ across the various operational sites. As the program attempts to define an agreed upon set of requirements, the user representatives change their requirements. In addition, an influential stakeholder has advocated the use of a new commercial-off-the-shelf (COTS) product as the solution approach.

As might be expected, there are incompatible scenarios that the new program would need to surface and explicitly address if it is to meet its various stakeholders' expectations. The first scenario reflects the expectation of one influential stakeholder who advocated the use of a COTS product that had been successfully used at one of the operational installations:

Stimulus	There is a desire to replace a complex component of a large legacy system with a COTS package
Environment	Within an established enterprise architecture with many local variations implemented that are largely different from each other
Response	The program runs a contest with a big prize to evaluate COTS packages for an enterprise-wide solution.

The second set of stakeholders, reflecting the operational users, is counting on the new system to quickly address their current needs. Understandably, these needs vary among the current fielded locations. During the time it takes the program to define an agreed upon set of requirements for each increment, the user representatives from the various fielded location change their requirements. This leads to the second acquisition scenario for this program:

Stimulus	Requirements for the next release keep changing
Environment	For a program with a fixed budget that must be carefully managed
Response	The program accepts the new requirements

Both of these scenarios are related to an acquisition quality attribute of flexibility. They describe how the program would accommodate different stakeholder needs. Unfortunately, the two scenarios are potentially incompatible with respect to designing the acquisition strategy. The first scenario is centered around the implementation of a common COTS product across all locations. This could provide sizeable value in terms of moving to one capability that is distributed across all fielded locations, but it may not meet what the current users consider urgent needs.

Implied in these two scenarios is a third set of stakeholders, the enterprise system engineers, who are advocating the implementation of an enterprise architecture that extends across all of the local

fielded implementations. This enterprise architecture could be incompatible with both of the above scenarios: each COTS product, by definition, is built to an architecture and a set of requirements that ABC has no control over. Further, the demands for local fielded implementations compete with architectural changes within a constrained budget.

Competition Between Acquisition and Architecture Scenarios

A different kind of incompatibility, and one less likely to be recognized, can occur between an acquisition-focused scenario and an architecture-focused one. One cause of this is that different communities (i.e., acquisition personnel and software personnel) and different sets of goals (i.e., business and mission) are involved in creating the scenarios.

In one program, for instance, organization XYZ had been under significant criticism for delay in responding to users in the field. A new director had been appointed with a mandate to remove bottlenecks and reduce the time between program start and initial operational capability.

The acquisition strategy therefore emphasized agility, responsiveness, and other such attributes. Among the elements of the strategy were several goals which (had the AQAW been available) could have led to appropriate acquisition quality attribute scenarios. For example, the goal of responsiveness led to a strategy of maximizing the use of open source software. If we were to couch that goal in terms of an acquisition quality attribute scenario, it might take the following form:

Stimulus	Users request significant new functionality to be delivered rapidly
Environment	during the program's development phase
Response	create the functionality rapidly by reusing open source and software from other projects to provide much of the capability

At the same time, however, the software architects had been warned that the situation in which the system was to be used made it necessary that the system was to be safety-critical and hard, real-time. Stringent certification standards would also apply to the system. For that reason, certification of the system would depend on removal of unreachable code from any reused or open source software. During a subsequent QAW, therefore, one of the key scenarios was aimed at a system/software quality attribute of certifiability:

Stimulus	A new requirement to adhere to a rigorous safety standard is applied to the system
Environment	during the program's development phase
Response	remove all unreachable code to insure that the system will pass stringent new certification standards

As in the previous example, both of these scenarios were well-intentioned, but they ultimately collided. Because, as the program unfolded, the open source that was most appropriate for the system had a considerable amount of unreachable code, the development underwent very large delays since the unreachable code was extensive, and was pervasive in all of the reused modules. The result was that the system was fielded almost three years late, since certification could not be done until the developers were convinced that all of the dead code was removed. By common agreement, the program office believed that while the open source software provided benefits, they were not as significant as expected.

In analyzing the 55 acquisition quality attribute scenarios generated from our interviews, we identified 24 scenarios as having a probable impact on the software architecture (these are noted in Appendix B). We would, therefore, expect to have one or more software quality attribute scenarios for each of the acquisition quality attribute scenarios that would need to be elicited, captured, and analyzed for potential incompatibilities. This will be an area of emphasis in phase three of our project.

3.5.6 Conclusions from Our Phase Two Research

Our effort during phase two was concentrated on understanding and demonstrating the applicability of acquisition quality attributes, particularly acquisition quality attribute scenarios. This follows the upper half of Figure 1, which emphasizes the relationships between the Business Goals, Acquisition Quality Attributes, and Acquisition Strategy entities. During our research in this phase, we captured and analyzed 75 acquisition quality attribute scenarios reflecting a dozen different acquisition quality attributes. As part of capturing the scenarios, we were able to show a critical link between the acquisition quality attribute and an acquisition strategy by first characterizing potential implications for an acquisition strategy and then identifying potential recommendations of how an acquisition strategy could accommodate the acquisition quality attribute scenario. We also had several colleagues with significant acquisition program office expertise review our acquisition quality attribute scenarios and associated acquisition strategy tactics to gauge the soundness of our results. They reported that the scenarios and acquisition tactics, such as the examples included with the scenarios listed in Appendices B and C, could surface issues and drive positive changes within a program's acquisition strategy, plans, and related artifacts.

We further showed that incompatibilities between scenarios, whether acquisition or software, can be distinguished and that the comparisons can surface issues important to a program. As a result, we have confidence that use of acquisition quality attribute scenarios are a viable path forward to aligning software architectures and acquisition strategies.

4 Summary and Proposed Future Steps

We are on a journey to provide a better approach to identify, understand, and reason about key drivers of a program's acquisition strategy and software architecture. Our research shows that as these drivers—in the form of business and mission goals—are either implicit or that conflicts exist among the goals which are not resolved. As a result, a program's acquisition strategy and software architecture are misaligned. This misalignment can lead to reduced operational capabilities and effectiveness, cost overruns, severe schedule slips, eventually resulting in systems failing to satisfy stakeholder needs. At worst, such misalignment can lead to program cancellations.

In making this pervasive problem tractable, we first created a model of the desired relationships among key entities—stakeholders, business goals, mission goals, acquisition strategy, software and system architecture, acquisition quality attributes, and software/system quality attributes. Forming the model then allowed us to (1) determine that there is no process for eliciting, capturing, and adjudicating the business goals of a program's stakeholders comparable to a process such as the DoD's Joint Capabilities Integration Development System (JCIDS) process; and (2) guide our research to prove our hypothesis on the existence and utility of acquisition quality attributes, embodied in the business goals, that drive the shape of the acquisition strategy—comparable to the relationship between mission goals, software/system quality attributes, and the software architecture.

Our research in phase two was on item 2 above (acquisition quality attributes) and is the focus of this report. We patterned our approach from that used by the SEI as it identified and codified key concepts and techniques around the relationship between software/system quality attributes and the software architecture. For the acquisition domain, we adapted the role of scenarios to create and demonstrate a viable way to express acquisition quality attributes specific to a particular program. We modified the original scenario elements in some ways: the acquisition strategy replaced the software architecture, the program replaced the system, and the program manager replaced the architect.

Underlying our work is the assertion that eliciting quality attributes so they can be analyzed is as critical for a sound acquisition strategy as it is for software/system architectures. We conducted various investigations in this regard, gaining experience within the acquisition domain and capturing numerous example scenarios. These investigations gave us the confidence that modifying the SEI QAW could be a viable starting point to elicit, capture, and analyze acquisition quality attributes and begin identifying potential impacts on an acquisition strategy. Our use of the prototype AQAW also indicated it is important to explore a more deterministic approach for eliciting acquisition quality attribute scenarios that cover the breadth of acquisition strategy drivers and is potentially less dependent on the particular participants attending a workshop. This will be an area of further investigation in phase three of our project.

4.1 Where We Go Next

We are now poised for the next critical stage in our journey—developing an alignment method. Our research confirms that alignment between the software and system architecture and acquisition strategy does not occur naturally, thus a method is needed to promote it. This alignment

method is intended to enable program managers to build acquisition strategies that more systematically eliminate one key cause of program failure—that of unintentionally over-constraining the software by not aligning the software architecture and the acquisition strategy. As a byproduct of using the method, a program manager should be able to express the steps taken and the tradeoffs made among business and mission goals to assure this alignment when asked to defend the acquisition strategy.

There are a number of operational and technical challenges that we will need to address as we move forward. First, the role of software in operational systems is now greater (often 70-80%, or more, of the operational functionality [Olagbemiro 2011]) and thus increasingly more critical to the success of a program. Along with this expanding role is a corresponding growth in priority risks related to both the system (e.g., achieving needed security, reliability, and adaptability) and programmatic aspects of the program (i.e., the ability of government, commercial, and contractor organizations to develop, deploy, and evolve systems reliably within acceptable cost, schedule, and performance constraints). Our alignment method will seek to bridge both types of risks.

Secondly, aspects of an acquisition that give rise to system and programmatic related risks typically emerge from different communities. Complex acquisition programs have diverse sets of stakeholders from these varying communities whose goals and priorities themselves may be misaligned. Operational users, combatant commanders, funding authorities, and acquisition team members often think they have the same priorities, but often in reality they do not. Too often, solutions are created based on the goals of one set of stakeholders, whose goals conflict with other stakeholders.

There are five gaps associated with the desired acquisition entity relationship model (see Figure 1) that our proposed alignment model will seek to remedy or at least, lessen their impact:

1. salient stakeholders are not readily identified and involved
2. acquisition quality attributes derived from business goals are absent
3. software/system quality attributes are not routinely used
4. quality attributes (acquisition and software/system qualities) are not used to inform acquisition strategies
5. acquisition strategies are not aligned with architectures

The technical basis for the alignment method is centered on extending and adapting concepts and methods that relate mission and business goals and software and system quality attributes to software and system architectures to now support the acquisition domain. Our starting point in developing an alignment method involves, at a minimum, the following aspects:

1. Selecting salient stakeholders: Adapt viewpoint-oriented methods, e.g., Controlled Requirements Expression (CoRE) [Mullery 1979, Finkelstein 1990] and stakeholder theory [Mitchell 1997]
2. Eliciting and capturing business goals: Adapt elements of the Pedigreed Attribute eLicitation Method (PALM) [Clements 2010] and integrate with elements of an acquisition-focused QAW
3. Eliciting and capturing acquisition quality attributes: Adapt quality attribute workshops (QAW) [Barbacci 2003] for the acquisition domain. Our investigation in modifying the

QAW to form a prototype Acquisition Quality Attribute Workshop (AQAW) showed great promise but we anticipate the need for further modifications.

4. Identifying and evaluating impacts to acquisition strategy: Adapt architecture tradeoff analysis method (ATAM) [Clements 2002] to evaluate acquisition strategies

Each of the contributing aspects noted above can be traced to its relevance in the acquisition entity relationship model as shown in Figure 2.

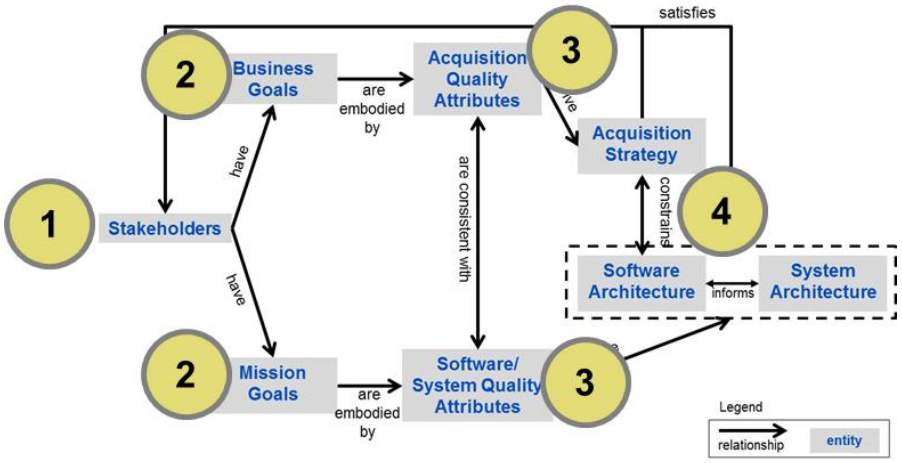

Figure 2: Contributing Methods and Techniques to the Proposed Alignment Method

Developing a useful alignment method will require more than simply linking the methods and techniques noted above. It will also need further work to refine our initial acquisition quality attribute taxonomy, government-oriented business goals, themes or groupings of acquisition quality attribute scenarios, and acquisition strategy tactics. Indeed, a particular issue we will face is whether to focus elements of the AQAW on eliciting goals using the categories identified by PALM (that are essentially stakeholder oriented) or some other organizing principle arising from the emerging themes on acquisition quality attribute scenarios. We will also need to leverage other research and methods from the broader architecture, requirements, and business communities.

Along with developing an alignment method, we also must take the critical step of piloting the emerging method with actual programs. We see several situations in which we could pilot parts or all of the method: as a program office is forming an acquisition strategy for a new program, for another major phase of a program, or for a major enhancement to an existing program. We are currently exploring several possible candidates for these tasks, and anticipate that at least some of them will be realized. As part of piloting the alignment method as a means to validate its utility, we anticipate applying elements of a case study approach [Yin 2009].

4.2 Final Thoughts

Our research to date has given us strong confirmation that our initial suppositions were sound, and that the method we will now develop will make a strong contribution to the acquisition community. In phase one of research, we saw ample evidence that, among the many pitfalls that plague acquisition programs, the lack of alignment between acquisition strategy and architectures ranked high on the scale of problems. During phase two, the gradual maturing of our concept of the acquisition quality attribute and the value of acquisition-related scenarios has taught us many con-

siderable lessons in the complex and subtle ways that acquisition strategy and architecture have mutual influence.

We thus look forward to phase three in which the real goal—an alignment method—is within our grasp. The path toward its development and validation will have its share of obstacles; but we are confident that it is an attainable goal, and one that will prove of value to a large number of professionals within the DoD and other government agencies.

Appendix A Initial List of Possible Acquisition Quality Attributes

The following is the initial list of possible acquisition quality attributes that we formed in phase two. These were derived from a combination of review of DoD acquisition strategy guidance and discussion with acquisition professionals, colleagues, and several brainstorming sessions within our team. We have refrained from definitions at this point in our work until there we have a larger set of acquisition scenarios. We currently subscribe to the tenet that precise definitions of quality attributes are only found within program-specific scenarios.

- Acceptability
- Accountability
- Affordability
- Appropriateness of contract
- Appropriateness of technology
- Achievability
- Accreditability
- Balance
- Commitability
- Communicability
- Competitiveness
- Contract manageability
- Credibility
- Effectiveness
- Evolvability
- Fairness
- Flexibility
- Implementability
- Legality
- Manageability of risk
- Management visibility
- Modifiability
- Promptness in reporting problems
- Responsibility
- Responsiveness
- Sensibility
- Staffability
- Suitability
- Sustainability
- Timeliness
- Traceability with requirements

Appendix B Acquisition Quality Attribute Scenarios Collected from Interviews

As discussed in the body of the report, much of our work in phase two involved collecting acquisition-related scenarios from as many sources as possible. This appendix lists the scenarios we gathered from our interviews with former program management office personnel and ITA members. These scenarios were generated by asking our interviewees to identify memorable negative and positive events that occurred in the programs with which they were associated. Thus, they gave us scenarios that largely represented dominant problems encountered in their programs.

This set of interview-derived scenarios represents many programs and as might be expected, there were repeating or at least similar, program events that gave rise to the collected scenarios. Similar scenarios were grouped together, forming the identified categories: capability of the industrial base, capability of the program office, sharing across different programs, use of new technologies, and the software lifecycle. This differs significantly from what we would expect when acquisition quality attribute scenarios are elicited for a single program. Here, we would anticipate that only one or two scenarios might be gathered for each of the categories noted above. This expectation is borne out in the results from the prototype AQAW.

In the tables in this appendix, we assigned a unique identifier to each scenario, associated each scenario with an acquisition quality attribute, and created one possible tactic that acquisition strategy developers might consider. We also denoted an acquisition quality attribute scenario that had probable software architecture implications with an asterisk (*) following the scenario identifier.

Note that the acquisition quality attribute scenarios listed here use the basic three-part form of scenarios expressed as sentences that concatenate the three parts. An actual program would then aggregate, consolidate, and prioritize these three-part scenarios and then expand them to a six-part form as described in Barbacci [Barbacci 2003].

Industrial Base Capability Scenarios

Table 3: Scenarios Related to the Capability of the Industrial Base

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
1	**Performability (of the contractor)**	One contractor is non performing due to lack of capability on their staff in a case where there are three parallel program offices competing for the same personnel; the contractors are encouraged to grow the workforce instead of poaching from each other.	Add language with respect to growing the workforce into the contracts.

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
2	**Performability (of the contractor)**	A contractor is missing early schedule milestones because of insufficient numbers of cleared personnel where the work they're performing is classified and it takes a long time to get people cleared; the program office tracks cleared personnel availability and suitability to do work prior to assigning the work.	(1) provide an opportunity for competing contractors to increase their pool of cleared people in advance of contract award, (2) require that contractors demonstrate that they have the cleared people necessary on their staff, (3) build clearance time lag into the program's early schedule.
3*****	**Performability (of the contractor)**	The contractor is falling behind schedule because they allocated work to a unit without prior domain expertise; the program office insists that the team that created the bid is the one that does the work.	Put restrictions in the contract (e.g., a key personnel clause) to ensure that the team that created the bid is the team that does the work.
4	**Performability (of the contractor)**	The schedule is not being met because of poor planning by a subcontractor that is developing a critical software component; the prime contractor terminates the agreement with the subcontractor and re-awards the work. Variant response: The prime contractor trains the subcontractor in project management.	Enforce the flow down of critical process maturity requirements from the prime contractor to ensure the sub-contractors have the planning skills they need and set aside funds to train/mitigate risks of gaps.
5	**Program Survivability**	A protest occurs claiming that one contractor had an unfair advantage shortly after the award of a new contract to update the airframe for a legacy program; the protest is dismissed because the source selection criteria show that there was no bias.	Establish source selection criteria such that bidders other than the incumbent are able to compete successfully.
6*****	**Flexibility**	A new need arises when we want to react quickly but there are only a limited number of contractors able to do the work; work to satisfy the need is added to an existing contract.	Award IDIQ contracts to multiple (recommend eight or so) vendors and issue task orders in a round-robin fashion.
7*****	**Affordability**	The program office finds that costs are growing and the cause can't be explained when running on a proprietary infrastructure; the program avoids vendor lock by switching to a non-proprietary infrastructure.	Put safeguards in place to avoid the vendor lock ensuring that there is always a viable competitor.
8	**Performability (of the contractor)**	It is determined that the integration contractor is too weak in a program where the government has awarded separate contracts to strong players for major components; the program office reallocates the integration tasks to someone else, ensuring that integration tasks are covered throughout the development.	Explicitly include the possibility of task reallocation in all contracts.
9	**Executability**	Contractor A is going to be replaced by contractor B in a program where contractor A has developed a complex system with massive amounts of data; all contractor A data is given to contractor B.	Pay close attention to intellectual property rights, ensuring that there are agreements for transfer of IP.
10	**Realism/ Executability**	One associate contractor refuses to share information with other contractors in an environment where the associate contractors compete with each other on other customer work; management structures and incentives are used to force collaboration.	Create contract requirements that allow the government to monitor collaboration.

Program Office Capability Scenarios

Table 4: Scenarios Associated with the Capabilities of the Program Office

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
11	**Performability (of the PMO)**	A hiring freeze inhibits hiring workers with skills needed to execute the program effectively while warfighters have urgent needs for changes to the system; the program plan is used to demonstrate the importance of the work and a waiver from the hiring freeze is obtained.	Have a staffing plan in place that includes specific required skills mapped to program activities and milestones; accompany this with the impact on the program should the skills be unavailable.
12	**Performability (of the PMO)**	A hiring freeze inhibits hiring workers with skills needed to execute the program effectively where warfighters have urgent needs for changes to the system; scarce resources are reallocated to the most pressing needs and routine work is postponed.	Put in place standardized processes to handle routine work so that scarce program office resources can be allocated to the most pressing needs.
13	**Performability (of the PMO)**	The program office personnel don't understand the program well enough to be effective in a program office with high personnel turnover; new personnel are presented with a guide to the program to bring them up to speed quickly.	Make new personnel effective as quickly as possible by having in place an organized way to transmit program knowledge that trades the overhead of documentation for the learning curve.
14	**Flexibility**	The program office receives a mandate to replace most of the PMO contractor personnel by personnel from an 8A/Veteran-owned company where the incumbents have been staffing most of the program functions; the existing contract is terminated cost effectively and the current contractor personnel transfer to the 8A company who wins the replacement contract.	Build termination potential into the contract.
15	**Realism**	The program office receives a mandate to replace most of the PMO contractor personnel by personnel from an 8A/Veteran-owned company where the incumbents have been staffing most of the program functions; the office is granted a waiver by demonstrating that no 8A company has the capability to execute the contract scope.	Track the personnel on the contract in order to be able to demonstrate the skills that will be lost.
16	**Realism**	There is a realization that the contractor is not meeting schedule for planned software drops because of the sudden appearance of new requirements where responsiveness to user needs is assuring the program of continued congressional support; the contractor is directed to increase the size of the workforce to accommodate the new requirements.	Build a suitable reward structure as an incentive for the contractor to be prepared to increase the size of the workforce.
17	**Flexibility**	There is a realization that the contractor is not meeting schedule for planned software drops because of the sudden appearance of new requirements where responsiveness to user needs is assuring the program of continued congressional support; schedules or priorities are changed for the planned drops.	Incorporate schedule flexibility into the program plan.

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
18	**Realism**	There is a realization that the contractor is not meeting schedule for planned software drops because of the sudden appearance of new requirements where responsiveness to user needs is assuring the program of continued congressional support; all new requirements except for absolutely "must haves" are rejected.	Incorporate the possibility of a severe guard on accepting new requirements into the program plan.
19	**Responsiveness**	The program office needs to get contractors working quickly even though it currently takes six months to award a new contract; instead of creating a new contract, the work is added to an existing contract.	Put in place a set of task order contracts that can be pre-awarded to qualified vendors.
20	**Responsiveness**	There is need for satellite coverage of a denied area in CENTCOM during an ongoing conflict; a new contract is awarded immediately and the satellite (with appropriate mission software) is built.	Possibly pre-award a number of contracts.
21*****	**Responsiveness**	There is a need to build satellites quickly where the satellites have to support at least seven different missions; the program decides to use pre-developed hardware and software components (akin to Lego) with a pre-awarded contract,	Award contracts aligned with major software segments so that changes can handled readily by different contractor teams.
22	**Responsiveness**	Lots of new requirements are coming from the operational users in the form of joint urgent operational needs (JUONS) to a program that takes over 400 days to be able to modify a contract; the program office allows the contractor to work at risk. Variant response #1: finds ways to accommodate new requirements without requiring contract changes. Variant response #2: finds out why it is taking 400 days to modify the contract and fix it.	Explicitly plan for incorporating new requirements without major modification to the contract. One alternative might be to include a priced contract line item that is structured as indefinite delivery/indefinite quantity (ID/IQ).
23*****	**Flexibility**	Requirements for the next release keep changing for the program with a fixed budget that must be carefully managed; the program accepts the new requirements. Variant response: reject requirements changes.	Plan in advance for dynamic requirements with changing missions; levy firm constraints on which requirements are accepted and which release will satisfy them; and establish a single, empowered authority that can accept or reject proposed requirements.
24	**Programmatic transparency**	The program office discovers that they don't have enough visibility into the products and data when building a system based on commercial solutions; a catalog of data and who needs it is developed and used in contract modification trades.	Plan (and fund) for a Commercial-off-the-Shelf (COTS) Management and Replacement Plan that addresses the COTS issues across the system lifecycle.
25	**Affordability**	Costs are growing and there is no explanation for the growth for a system being developed that is running on a proprietary infrastructure; analysis of developer-provided metrics is initiated to explain cost increases.	Put safeguards in place to require appropriate controls (and metrics) to manage cost increases to ensure that suitable metrics and risk mitigation is in place.

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
26	**Flexibility/ Executability**	There is a mandate that all contracts must be executed as if they are firm fixed price (FFP) when the contract is currently for time and materials; a ceiling on time is established so that the time and materials contract can behave as if it is FFP.	Insert language or metrics to demonstrate that FFP tasks are definable on time and materials contracts. If not definable, then know the limitations that force a change to the new contract type.

Sharing Across Programs Scenarios

Table 5: Scenarios Associated with Sharing Across Different Programs

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
27*	**Realism/ Flexibility**	It is determined that it has become too complex to manage the planned product line strategy because, for example, the customer expectations are too different or divergent in spite of much pressure to build common ground stations; work is stopped until the product line strategy is worked out and is executable.	Focus on necessary activities that will develop the product line as a viable strategy.
28*	**Realism/ Flexibility**	It is determined that it has become too complex to manage the planned product line strategy because, for example, the customer expectations are too different or divergent in spite of much pressure to build common ground stations; systems for the different customers are allowed to diverge for the short term but with a requirement to show how they will converge at a later stage.	Create a strategy whereby, even if the two variants can't come from a product line now, there is a fallback strategy where they can begin sharing later—e.g., by sharing something (platform, architecture, components) now, and plan for flexibility to accommodate cost and schedule impacts and define how developers would be incentivized to follow the strategy.
29*	**Flexibility**	A change in the environment is putting greater pressure on sites to share data with each other where originally each site had the freedom to implement changes as they wished and those changes are now inhibiting data sharing; all variations of the system are removed.	Establish a single steering group with responsibility for increasing site sharing. Then stamp out variants by developing a strategy for consolidating instances (make regional instances first). Enable this by centralizing funding and development: drive everyone to a common goal to improve data sharing (and stop funding individual variations).
30*	**Realism**	There is pressure to use non-developmental item (NDI) products that are incompatible with the desired architectural strategy of creating an "Apple-like" system; the pressure is ignored and the program continues to follow and enforce their architectural strategy.	Do more upfront work in terms of "market research" so that the approach can be explained and defended against alternatives through the life of the system; this research needs to be done in the context of the enterprise strategy.

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
31*	**Realism**	There is pressure to use NDI products that are incompatible with the desired architectural strategy of creating an "Apple-like" system; the program requests a waiver from the architecture strategy.	Do more upfront work in terms of "market research" so that the approach can be explained and defended against alternatives through the life of the system; this research needs to be done in the context of the enterprise strategy.
32*	**Realism**	The government furnished equipment is not available when needed in part because there was no upfront commitment to provide resources to create it even though the government furnished equipment (GFE) has been mandated for use by all programs; the program proceeds using an alternate to the GFE.	Factor in the preliminary development of an equivalent of the GFE into the program's budget and schedule.
33*	**Realism**	The program discovers that it cannot fully migrate to the organization's common processing infrastructure (because of program performance impacts, costs, …) where the common processing infrastructure is mandated in order for the enterprise to meet emerging needs without the need for new dedicated resources; the legacy programs are migrated regardless of objections. Variant response: establish a waiver process.	Explicitly plan for migration of legacy (which applications will migrate and when) in implementing the common processing infrastructure. Waiver processes should be in place and well understood.
34	**Survivability**	A program wants to build a new message format for themselves where every program has been directed to solve multiple program needs even though every system is used to solving problems to their specific need; the outlying programs are terminated.	Appeal to management senior enough to stop the renegade program and enforce the requirement that all programs are required to use a common solution.
35*	**Executability**	New satellite specifications are not available as needed to support the terminal development where the satellite and terminal are being developed concurrently; Technical interchange meetings to enhance situational awareness are held regularly between the satellite and terminal contractors.	Require, through language in the contracts, a monthly/quarterly meeting to assess technical issues between the contractors (and with any other parallel programs).
36*	**Executability**	Contractor to contractor deliveries are not being made on time where there are multiple contractors with complex interdependencies among the contracts; Technical interchange meetings to enhance situational awareness are held regularly between the satellite and terminal contractors.	Require, through language in the contracts, a monthly/quarterly meeting to assess technical issues between the contractors (and with any other parallel programs).
37	**Survivability**	There is a growing fear that political forces in the Services will kill the program for a program that was going to be highly controversial with the Services because of the effect it would have on each Service's business practices; Ensure that the program has high level visibility to keep the political forces in line.	Develop a strategy for keeping senior leadership aware of the program. This could include a senior steering group where issues between services can be vetted and resolved.
38	**Flexibility**	Reliability issues are discovered where all previous emphasis had been on functional performance; the program office adds a system of systems engineer with control over the programs; documents the system architecture and assigns an architect.	It is important to protect system of system engineering over the lifecycle so that it is possible to perform root cause analysis and allocate the responsibility to fix it.

Innovative Solution or New Technology Scenarios

Table 6: Scenarios Associated with New Technology or Other Innovations

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
39*	**Flexibility**	One or more of the new approaches is found not to work where the program intends to emphasize the use of new technology to support a new approach; the failing approach is dropped and emphasis is place on the remaining viable approaches.	Keep the maximum number of independent/Innovative contractor solutions in play until the program can understand their feasibility.
40*	**Affordability**	The cost of maintaining parallel approaches is no longer acceptable where multiple developments are run in parallel for purposes of risk reduction; a down-select is performed based on a bake-off between the current developments.	Keep the maximum number of independent/Innovative contractor solutions in play until the program can understand their feasibility.
41	**Transparency**	Questions are raised about the absence of a Critical Design Review (CDR) for a program where the contractor is using Agile development; The emerging, though fragmented, documentation from the Agile processes is used in lieu of formal CDR documentation to prove that a traditional CDR isn't needed.	Keep documentation on Agile methods that shows the use of periodic development reviews as a reasonable incremental CDR. Should also include an event to make sure the contractor processes are understood – and compatible with government needs.
42	**Affordability**	Money is running out because the true cost of building a new system is higher than expected where the system has a high profile and will be jointly operated by two different government agencies; fallback strategies are employed that meet the most compelling needs with the available funds Variant Response: work on the new system is terminated and interfaces between legacy systems are constructed; Variant Response: commercial options are investigated.	Continually look for alternate courses of action should the current course fail.
43*	**Affordability**	There is a desire to install a new capability on a set of platforms with many variants where the variants are widely divergent and use different programming languages (even assembler); the cost vs. benefit for each variant is examined so that a conscious decision can be made on which variants will receive the new capability.	Put in place a no-penalty clause that permits some variants to not get the capability if the cost is prohibitive.
44	**Realism**	A team from another program proposes developing an extension of their ground station where the program manager has already established a team within the program to build the new ground station; the PM allows both developments to proceed, performing continuous evaluations until a clear winner emerges.	Encourage competition between teams, the strategy allows for continuous evaluations and decisions on the role of emerging, competing technologies.
45	**Survivability**	A team from another program proposes developing an extension of their ground station where the program manager has already established a team within the program to build the new ground station; the program manager enforces the decision to use new technologies for the new ground station approach.	Apply more work up front in "market research" so that the approach can be explained and defended against competitors.

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
46*	**Innovativeness/ Flexibility**	There is a desire to replace a complex component of a large legacy system with a COTS package within an established enterprise architecture with many local variations implemented that are largely different from each other; the program runs a contest with a big prize to evaluate COTS packages for an enterprise-wide solution.	Conduct (and fund for) market research through the life of the program that will make the program office aware of commercial opportunities. Based on this market research, be prepared to incentivize high-payoff commercial products.

Software Lifecycle Scenarios

Table 7: Scenarios Associated with the Software Lifecycle

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
47	**Flexibility**	The program office loses some funding in a challenging budget environment; the office adjusts program requirements and costs across the contractor base.	The contracts and technical strategy need language that allows the program office to adjust funding.
48*	**Flexibility**	A budget cut occurs in the program where there is high probability of financial instability for a multi-segment system; some of the work is terminated to maintain the program schedule.	Require a path to completion in an unstable funding environment.
49	**Performability (by the PMO)**	The program office loses control of the system configuration where the software is simultaneously being developed and maintained by separate organizations; The program office sets up a version control mechanism that is enforced; work is serialized; and publication of what changes are being made, and by whom, is enforced.	Have clear configuration management and change control with a clear allocation of responsibilities, including coordination and transfer of data.
50	**Affordability**	The program wants to avoid uncontrolled costs even though they want to build an unprecedented capability (with significant technical risk and unknown requirements); the program requires the contractor to design their implementation based on an approved prototype.	Use government labs to produce the first article and tightly control the engineering phases.
51	**Flexibility**	The operational flight dynamics have placed unexpected demands on the system where the production of hardware is already underway and has become too hard to change; the program makes modifications to the software to fix the system problems.	Use a proactive strategy that anticipates software changes throughout the life cycle.
52*	**Flexibility**	There is a demand for early delivery of partial operational capability even though the complete operational software is not yet ready; the requirements are segregated and the architecture is designed for incremental delivery with the essential capabilities built first.	Build in a contingency plan that allows for early deployment of partial capability. This could be through the use of an agile development process that is continuously building capabilities on an operational-quality code base.

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
53	**Affordability**	The program office fears the cost of maintaining low quality software where the office understands that maintenance costs are driven by both software quality and complexity; the development contract is required to include FFP maintenance before development begins.	Find ways to incentivize the development contractor to care about long term maintenance costs.
54*****	**Survivability**	A campaign is being mounted to kill the program where there is lots of competition from other programs for scarce funding; the program is extended to include joint constituencies.	Make sure the program is joint and able to add new constituencies/advocates – sponsored by a unified commander. Use of product line (or some of the product line principles) could allow for more sharing and makes clearer the effects of new customer requirements.
55	**Survivability**	A formerly quick reaction capability (QRC) program is threatened with termination where the program has transferred from reporting directly to the CIO to another group (that is at least three levels down); the benefits of the QRC approach in limited circumstances are readily available.	Use continuous demonstrations in order to prove the value of the QRC approach.

Appendix C Acquisition Quality Attribute Scenarios Collected from Acquisition QAW (AQAW)

We prototyped a Quality Attribute Workshop adapted for the acquisition domain, which we refer to as an Acquisition Quality Attribute Workshop (AQAW), on a program currently supported by the SEI that was working toward the release of a request for proposal (RFP). We asked SEI staff, who were providing technical support, to play the part of key acquisition personnel. The scenarios captured in the workshop differ from those listed in Appendix B. In the workshop we worked with a single program and we explicitly elicited and captured scenarios that were driven from their specific business goals. Thus, there are at most two scenarios representative of each of the scenario categories noted in Appendix B. This is what we would expect in a typical AQAW.

The following are the unprioritized scenarios created as part of the AQAW using the basic three-part form of scenarios expressed as sentences that concatenate the three parts. For this prototype workshop due to time constraints, we did not perform the aggregation, consolidation, prioritization, and refinement steps found in a complete QAW (as described in Barbacci [Barbacci 2003]). The scenario identifier represents the order in which the scenarios were generated during the workshop. Where we identified a possible impact on the software architecture, we appended an asterisk (*) to the scenario identifier. In addition, we created a potential acquisition tactic for each scenario as an example of what acquisition strategy developers might consider.

Table 8: Scenarios Captured in Prototype AQAW

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
1	**Flexibility**	The user's system requirements change radically 30 days before the RFP is released when the "go live" date is fixed; the RFP is released regardless.	Establish fallback strategies that protect the "go live" date.
2*****	**Affordability**	We discover that the cost of operating the system will be higher than the ceiling mandates during development but before initial fielding; the system (including its architecture) is shifted to a less costly alternative.	Emphasize the need for architecture adaptability and flexibility.
3*****	**Affordability**	The pricing strategy of the database vendor changes during development becoming prohibitively expensive; the system is switched to use a different database.	Solicit database analysis and demonstrations to understand technical arguments in the response to the RFP.
4*****	**Performability (of the system)**	Data migration between database products or data models from the legacy to the new system is found to be impossible during development; the program office can #1: back out of any new schema and adopt the schema from the legacy system or #2: know from early experiments that migration is feasible.	Validate that data migration is feasible and doesn't affect performance.

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
5*	**Flexibility**	An external event triggers a need for immediate (within one year) fielding of capability and the legacy system cannot meet the need during system development; a vendor supplied (COTS-based) capability is fielded with the understanding that vendor lock in may have been created.	Encourage iterative development or active monitoring and participation in relevant market segments.
6*	**Flexibility**	A new policy mandate to use a government platform-as-a-service (PaaS) is released after development starts; the solution under development is ported to the new platform.	Implement a PaaS strategy that would allow for migration to a new platform. Make the PaaS a separate line item and require a demonstration that the solution can be ported to another platform.
7	**Performability (of the contractor)**	The new contractor starts behaving in a fashion the program office doesn't like during development; the government exercises its contractual rights and imposes a suitable style of behavior.	Create performance measures of and incentives for desired behavior.
8	**Flexibility**	After contract award, the contractor is unable to provide unlimited (or government purpose) data rights for all system elements even though the government needs were stated in the RFP; the government and contractor follow a predetermined process for mediating the dispute.	Ensure that the contract and RFP defines appropriate rights and responsibilities with mediation mechanisms and penalties clearly defined.
9	**Performability (of the contractor)**	We find, during source selection, that the offeror's solution uses commercial products where government purpose rights are not obtainable; the offeror is not selected based on the RFP selection criteria.	Call out data rights as part of the selection criteria.
10	**Flexibility**	The government discovers that it does not have the data rights to fulfill one of its responsibilities after the contract has been awarded; the contract is either terminated for cause or the government demands a previously agreed penalty.	Surface potential gaps and define appropriate responsibilities, mediations, and penalties before contract award, perhaps using a request for information.
11	**Performability (of the contractor)**	The program office discovers that it does not have needed experience or skills midway during development; the program hires (or contracts for) the skills it needs.	Maintain a staffing plan showing the skills required to oversee all aspects of the contract and that there is a plan to hire or develop the needed skills—with an understanding of the impact of not being fully staffed.
12	**Affordability**	Costs start to overrun because the contractor design has diverged from the basis of estimate (the "to be" vision) where the "to be" concept/engineering vision is not reflected in the acquisition strategy; work is redirected to follow the original "to be" vision.	Ensure that the contract requires alignment to the vision and puts in place frequent reviews to ensure that no divergence occurs.
13*	**Realism**	The contractor asserts that to make progress with their bid, they need an explicit description of the target platform where the system requirements are not fully defined and there has been no discussion on the establishment and negotiation of service level agreements; the government either #1: lets the contractor select the platform or #2: makes a platform choice quickly.	Ensure that whichever platform the government chooses is clearly defined in the RFP.

ID	Acquisition Quality Attribute	Scenario	Potential Acquisition Tactic
14*	**Schedulability**	It is found that the certification to operate process is taking more time than scheduled during the middle of certification; the system is broken into smaller pieces each of which can be given accreditation separately.	Integrate IA personnel and considerations into the development process—also provide early deliverables to IA.
15	**Performability (of the system)**	The acquisition authorities decide that they will not allow the system to operate in a critical environment during the certification process; a workaround is developed for the critical environment.	Integrate information assurance (IA) personnel and considerations into the development process—also provide early deliverables to IA.
16	**Flexibility**	A government engineering center (GEC) steps in and claims the right for system integration either before or during source selection; allow the GEC to submit a proposal.	Establish criteria permitting a government organization to compete fairly.
17*	**Flexibility**	Public debate on privacy triggers the addition of more constraints on the type of data that is considered private information or some new type of sensitive data is added during development; the system is modified to accommodate the new reality.	Plan to incorporate new types of data and be explicit about both privacy and performance considerations.
18	**Performability (of the system)**	The program discovers that the system is not going to meet performance parameters or scale to meet the system requirements late in development or in system test; performance requirements are scaled back for now and deferred to the next iteration.	Maintain a continuing relationship with the stakeholders to allow for adjustments to performance requirements.
19	**Flexibility/ Realism**	The program discovers that the system is not going to meet performance parameters or scale to meet the system requirements late in development or in system test; performance requirements are scaled back for now and deferred to the next iteration.	Maintain a continuing relationship with the stakeholders to allow for adjustments to performance requirements.
20*	**Realism**	During development, the classification for the new system becomes SECRET even though the legacy system was unclassified, breaking the interaction with other government systems; system development proceeds without the other government systems.	Let this be a problem for the other government agencies and not belong to the program.

References/Bibliography

URLs are valid as of the publication date of this document.

[Alexander 1977]
Alexander, Christopher, Ishikawa, Sara, & Silverstein, Murray. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977 (ISBN 0-19-501919-9).

[Barbacci 2001]
Barbacci, Mario, Ellison, Robert, Stafford, Judith, Weinstock, Charles, & Wood, William. Quality Attribute Workshops (CMU/SEI-2001-TR-010). Software Engineering Institute, Carnegie Mellon University, 2001. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5613

[Barbacci 2003]
Barbacci, Mario, Ellison, Robert, Lattanze, Anthony, Stafford, Judith, Weinstock, Charles, & Wood, William. *Quality Attribute Workshops* (QAWs), *Third Edition* (CMU/SEI-2003-TR-016). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6687

[Bass 2012]
Bass, Len, Clements, Paul, & Kazman, Rick. *Software Architecture in Practice, 3rd Edition*. Addison-Wesley, 2012.

[Baxter 2008]
Baxter, Pamela & Jack, Susan. "Qualitative Case Study Methodology: Study Design and Implementation for Novice Researchers." *The Qualitative Report 13*, 4 (December 2008); 544-559 http://www.nova.edu/ssss/QR/QR13-4/baxter.pdf

[Blanchette 2010]
Blanchette, Stephen, & Bergey, John. "The Chief Software Architect in U.S. Army Acquisition." *CrossTalk* (November/December 2010).

[Brown 1998]
Brown, W.J. *Antipatterns: Refactoring Software, Architecture, and Projects in Crisis*. Wiley and Sons, 1998.

[Brownsword 2013]
Brownsword, Lisa, Albert, Cecilia, Carney, David, Place, Patrick, Hammons, Charles (Bud), & Hudak, John. *Isolating Patterns of Failure in Department of Defense Acquisition* (CMU/SEI-2013-TN-014). Software Engineering Institute, Carnegie Mellon University, 2013. http://www.sei.cmu.edu/library/abstracts/reports/13tn014.cfm

[Charette 2003]
Charette, Robert, McGarry, John, & Baldwin, Kristen. *Tri-Service Assessment Initiative Phase 2 Systemic Analysis Results*. 2003.

[Clements 2002]
Clements, Paul, Kazman, Rick, & Klein, Mark. *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley, 2001 (ISBN-10: 0-201-70482-X; ISBN-13: 978-0-201-70482-2).

[Clements 2010]
Clements, Paul & Bass, Len. *Relating Business Goals to Architecturally Significant Requirements for Software Systems* (CMU/SEI-2010-TN-018). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9347

[Conway 1968]
Conway, M.E., "How do Committees Invent." *Datamation*, 14(5):28-31, 1968.

[DAU 2011]
Defense Acquisition University. *Glossary of Defense Acquisition Acronyms and Terms*. 14th Edition, 2011.

[Finkelstein 1990]
Finkelstein, Anthony, Kramer, Jeff, & Goedicke, Michael. "ViewPoint Oriented Software Development." *Proceedings of Third International Workshop on Software Engineering and its Applications*, Toulouse, December 1990.

[Firesmith 2008]
Firesmith, Donald. *The Method Framework for Engineering System Architectures*, Auerbach Publication, 2008 (ISBN 978-1-4200-8575-4). With Peter Capell, Dietrich Falkenthal, Charles B. Hammons, DeWitt T. Latimer IV, and Tom Merendino.

[Henderson 1993]
Henderson, J.C. & Venkatraman, N. "Strategic Alignment: Leveraging Information Technology for Transforming Organizations." *IBM Systems Journal 32*, 1 (1993).

[IEEE 1998]
IEEE 1220-1998, IEEE Standard for Application and Management of the Systems Engineering Process.

[Klein 2010]
Klein, John & Gagliardi, Michael. *A Workshop on Analysis and Evaluation of Enterprise Architectures* (CMU/SEI-2010-TN-023). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9363

[MacCormack 2011]
MacCormack, Alan, Rusnak, John, & Baldwin, Carliss, "Exploring the Duality between Product and Organizational Architectures: A Test of the "Mirroring" Hypothesis." Harvard Business School, 08-039m, 2011

[Mitchel 1997]
Mitchell, R., Agle, B., Wood, D. "Toward a Theory of Stakeholder Identification and Salience: Defining the Principle of Who and What Really Counts." *The Academy of Management Review,*

22, 4 (Oct., 1997); 853-886
http://www.jstor.org/stable/259247

[Mullery 1979]
Mullery, G.P. *CORE - A Method for Controlled Requirement Specification*, CHI479-5/79/0000-0126500.75. IEEE 1979.
http://ss.hnu.cn/oylb/tsp/CORE-mullery.pdf

[Olagbemiro 2011]
Olagbemiro, Albert, Mun, Johnathan, Shing, Man-Tak, *Applications of Real Options Theory to DoD Software Acquisitions*, Defense Acquisition Research Journal, Defense Acquisition University, January 2011.
http://www.dau.mil/pubscats/PubsCats/AR%20Journal/arj57/Olagbemiro_ARJ57.pdf

[OPF 2009]
Open Process Framework (OPF)
http://www.opfro.org/index.html?Components/WorkProducts/ArchitectureSet/Architectures/Architectures.html~Contents

[Stake 1995]
Robert E. Stake, *The Art of Case Study Research.* Sage, 1995 (ISBN 0-8039-5767-X).

[Strassman 1998]
Strassmann, Paul A. *What is Alignment? Alignment is The Delivery of the Required Results.*1998.
http://www.strassmann.com/pubs/alignment/

[USD 2012]
Under Secretary of Defense Memorandum to the Defense Acquisition Workforce, Better Buying Power 2.0: Continuing the Pursuit for Greater Efficiency and Productivity in Defense Spending. November 13, 2012.

[Yin 2009]
Robert K. Yin. *Case Study Research: Design and Methods*. Fourth Edition. SAGE Publications. California, 2009 (SBN 978-1-4129-6099-1).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	February 2014	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Results in Relating Quality Attributes to Acquisition Strategies	FA8721-05-C-0003

6. AUTHOR(S)

Lisa Brownsword
Cecilia Albert
David Carney
Patrick Place

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2013-TN-026

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFLCMC/PZE/Hanscom Enterprise Acquisition Division 20 Schilling Circle Building 1305 Hanscom AFB, MA 01731-2116	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

In the acquisition of a software-intensive system, the relationship between the software architecture and the acquisition strategy is typically not specifically examined. The first phase of our research discovered an initial set of failure patterns that result when these two entities become misaligned. Programs with these failure patterns experienced reduced operational capabilities and effectiveness, cost overruns, and significant schedule slips. In other words, these programs resulted in systems failing to satisfy stakeholder needs.

This report describes the conceptual foundations for our project and summarizes the first phase as context for the second phase, which is the major thrust of this report. The current research has centered on demonstrating the existence and utility of acquisition-related quality attributes, embodied in a program's business goals, which then drive the shape of the acquisition strategy. This is comparable to the relationship between mission goals, software-related quality attributes, and the software architecture. This report describes the approach used in phase two to generate 75 acquisition-related quality attribute scenarios based on data derived from more than 23 large government programs spanning business, logistics, command and control, and satellite domains.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Acquisition, software architecture, quality attributes, business goals	56

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL